Project Recap & Lessons Learned

Weather Data Analysis

Initial Hypothesis:

I hypothesize that there exists a correlation between temperature and humidity, where higher temperatures are associated with higher humidity levels. Additionally, I expect to build a model with good accuracy which predict the daily summary.

In this project, I examined how various factors influence the weather around the world. The factors I chose to analyse are temperature, humidity, wind direction, visibility, wind speed, precipitation and other relevant factors. Another focused area in this project is to analyse the trends and patterns in weather data to understand how various meteoritical factors influence each other and impact weather conditions.

Data Acquisition and Data Pre-processing:

The dataset "weatherHistory" is taken from Kaggle website [1]. One well-known platform in the data science space is Kaggle. It offers a venue for data scientists, machine learning enthusiasts, and practitioners to compete, interact, learn, and exchange, according to their needs.

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

Table 1: Sample data (Jupyter Notebook)

The above table is created in jupyter notebook using panda's library [2]. From the above table, I can see that the dataset contains 12 columns in which 8 of them are decimal type such as Temperature, Apparent temperature, Humidity, Wind Speed, Wind Bearing, Visibility, Loud Cover, Pressure, 3 of them are string type such as Summar, Precip Type, Daily Summary, one is Date Time type which is names as Formatted Date.

The dataset contains 96453 entries, each of them are recorded on different time and date. Using this dataset I am going to analyse the required attributes, build a model to predict the daily summary of weather and complete the hypothesis. The below table is plotted using pandas [2] which describes the decimal attribute's mean value, standard deviation, minimum value, maximum value and more.

```
data.describe()
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000

Table 2: Data Description (Jupyter Notebook)

Data pre-processing plays a crucial role in analysing the data, creating ML models. In the process of data cleaning, I used some techniques like finding missing values, eliminating duplicates, changing date format, looking at the outliers. By using the techniques, I eliminated the unwanted data. At last, the remining entries are 92817.

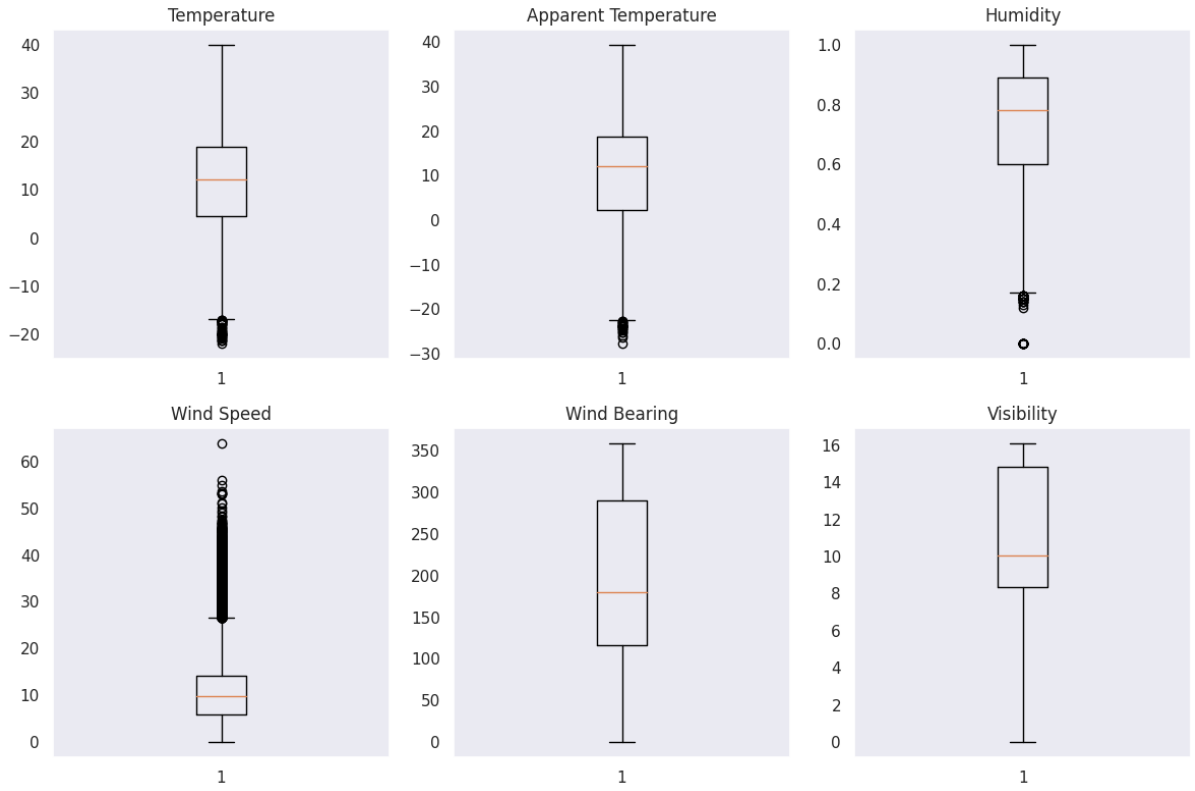

Figure 1: Outliers (Jupyter Notebook)

The above figure 1 is plotted using matplotlib library [3]. The outliers can be identified using box plots of required attribute. A total of 3118 entries are deleted as outliers.

Another important part of data pre-processing is converting the data into required useful way. I converted columns into required categories such as temperature as numeric, summary as categorical, time as discrete. Later I examined the summery column. I found 14 different summaries. Among those 14 summaries I chose 3 to examine ‘overcast’, ‘clear’, ‘foggy’. Finally, I left with 33840 entries to explore.

Platform used and libraries:

- Platform used: Jupyter Notebook, Project Jupyter's web-based user interface is called Jupyter Notebook [4]. In essence, it's an interactive computing platform utilized mostly in computational journalism, machine learning, data science, and scientific computing.
- Software used: Anaconda [5], For the Python and R programming languages, Anaconda is a free and open-source software distribution that makes package management and deployment easier. I used anaconda software to access Jupyter Notebook.
- Libraries:
    1. Pandas: [2] An effective library for data analysis and manipulation that provides data structures like Series and DataFrames for managing tabular data.
    2. Matplotlib: [3] An essential Python visualization package that may be used to create interactive, animated, and static graphics.
    3. Seaborn: [6] An advanced interface for producing aesthetically pleasing statistical visualizations that is built over top of matplotlib.
    4. Numpy: [7] The core of Python's numerical computation, including strong matrix and array operations for scientific computing.
    5. Datetime: [8] An necessary library for manipulating and evaluating time-based data is one that deals with dates, times, and timedeltas.
    6. Joblib: [9] Created to provide Python objects permanence, enabling the saving and loading of other objects, including machine learning models.
    7. Sklearn: [10] A well-known machine learning library that provides an extensive collection of tools and algorithms for different machine learning applications.
    8. Tensorflow: [11] Strong open-source framework for large-scale machine learning and numerical computing that is especially well-suited for deep learning applications.
    9. Wordcloud: [12] A library that creates word clouds from text data to show word frequencies graphically.

Exploratory data analysis:

In the part of data exploration, I plotted various graphs to analyse the data.

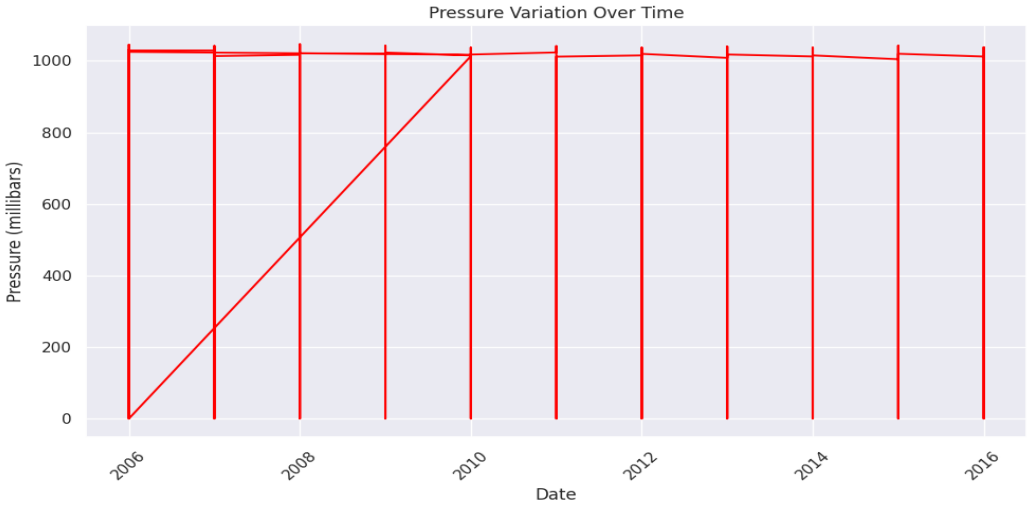

Figure 2: Pressure variation over time (Jupyter Notebook)

The figure 2 is plotted using matplotlib library [3]. The "Date" x-axis spans the years 2006 through 2016. "Pressure (millibars)" is the label on the y-axis, which runs from 0 to 1000. A line plot of pressure over time is displayed on the graph.

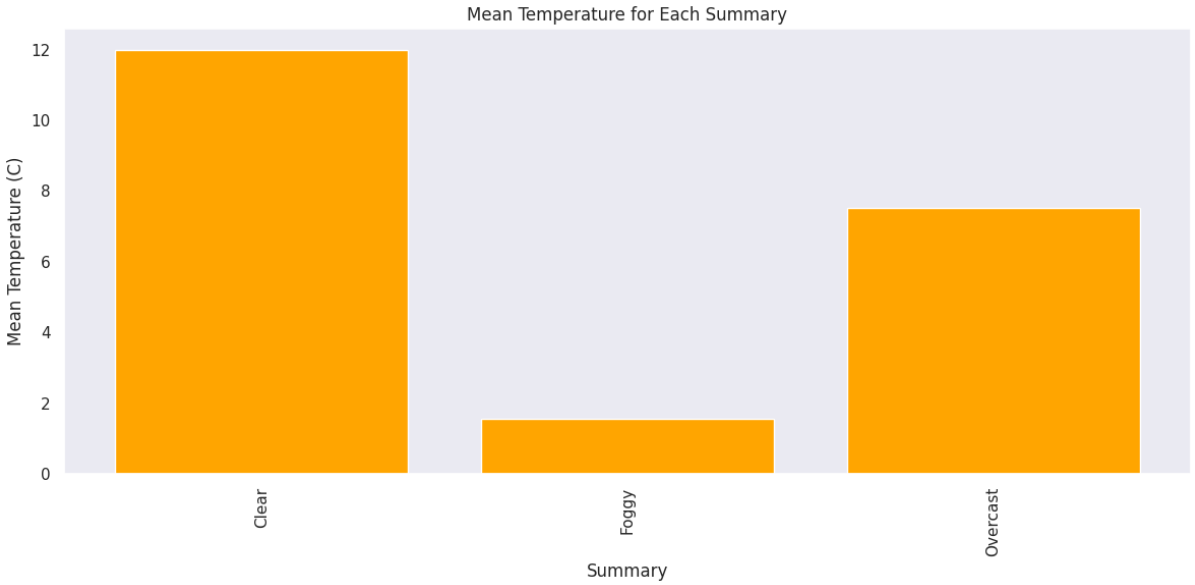

Figure 3: Mean Temperature for each summary (Jupyter Notebook)

The figure 3 is plotted using matplotlib [3]. The "Mean Temperature for Each Summary" bar graph is the one displayed. The three classifications "Clear", "Foggy", and "Overcast" are shown on the x-axis. The mean temperature in degrees Celsius is displayed on the y-axis. According to this graph, temperature and cloud cover are positively correlated. Days with clear skies and little to no cloud cover had the highest average temperature. On the other hand, days that are cloudy and foggy tend to have mean temperatures that are lower.

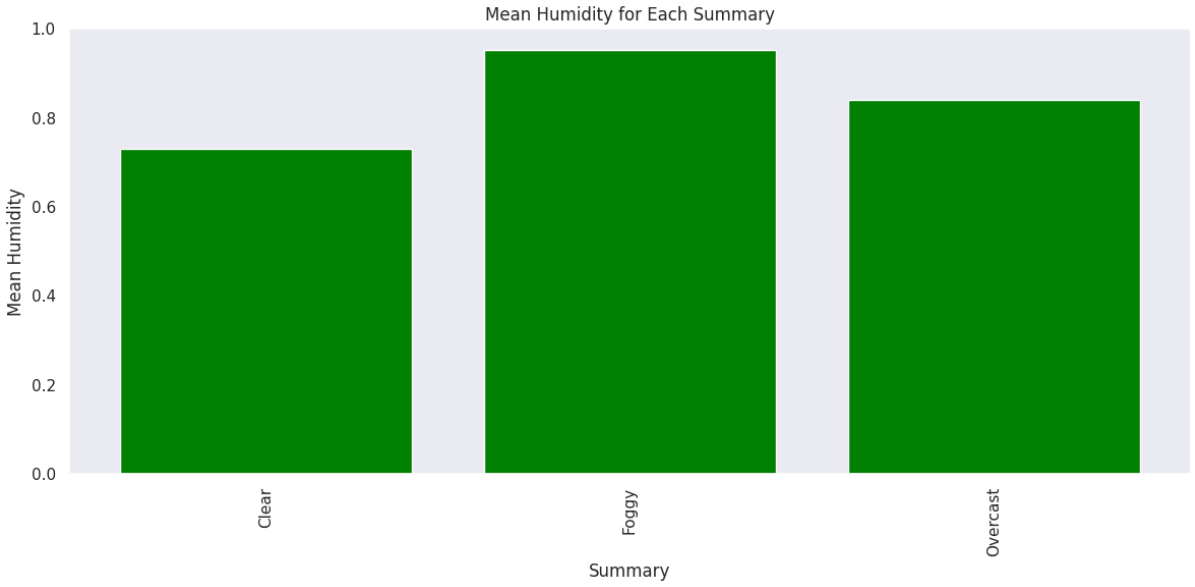

Figure 4: Mean humidity for each summary. (Jupyter Notebook)

The figure 4 is plotted using matplotlib [3]. The graph displays the average humidity for a number of different weather summaries. "Clear", "Foggy", and "Overcast" are the three

classifications shown under the "Summary" x-axis label. "Mean Humidity" is the label on the y-axis, and its values span from 0 to 1.0. For every category in the weather report, bars are shown on the graph. This graph indicates that humidity and cloud cover are inversely correlated. The days with the lowest mean humidity are clear ones with little to no cloud cover. On the other hand, days that are cloudy and foggy tend to have greater mean humidity.

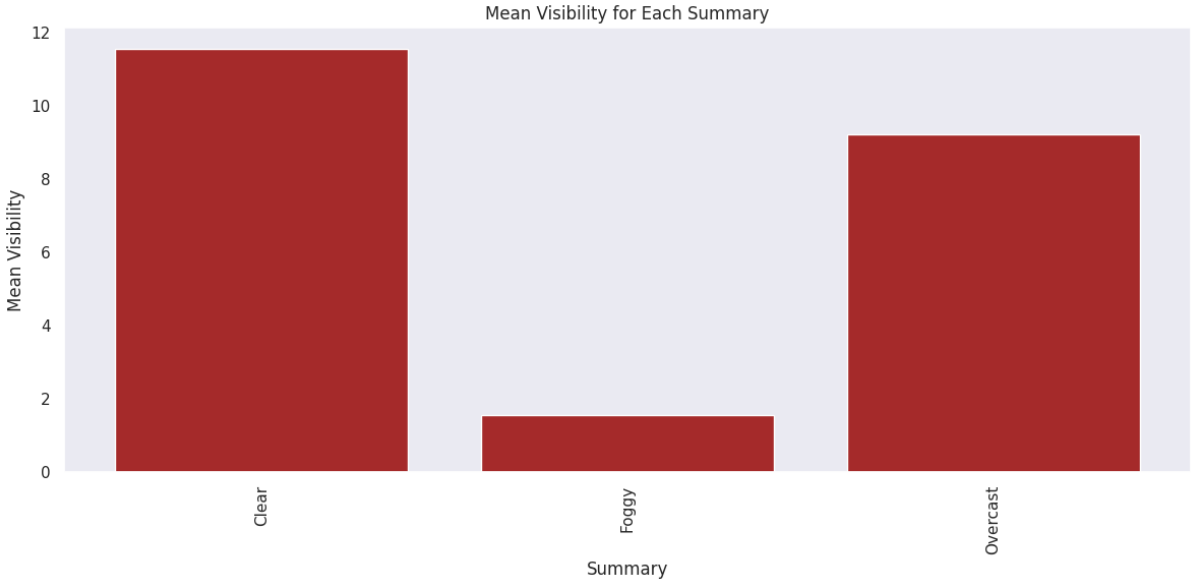

Figure 5: Mean visibility for each summary (Jupyter Notebook)

The figure 5 is plotted using matplotlib [3]. The graph displays each summary statistic's mean visibility. "Clear", "Foggy", and "Overcast" are the three classifications shown under the "Summary" x-axis label. The scale runs from 0 to 12, with the values not displayed on the "Mean Visibility" y-axis label. For every category in the weather report, bars are shown on the graph. This graph indicates that visibility and cloud cover are positively correlated. The maximum mean visibility occurs on clear days with little to no cloud cover. On the other hand, days with the highest amount of cloud cover and fog had the lowest mean visibility. Days with clouds have a mean visibility that is in between.

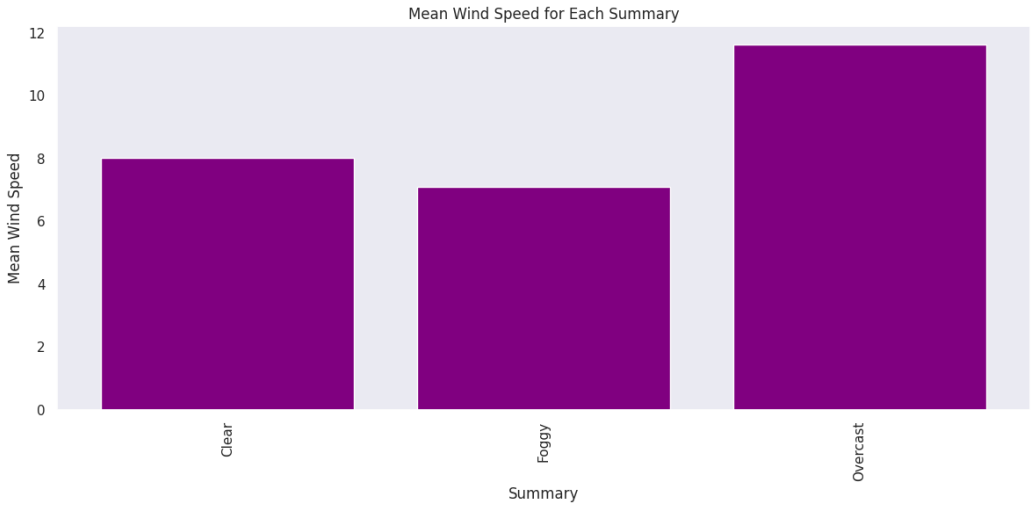

Figure 6: Mean wind speed for each summary (Jupyter Notebook)

The figure 6 is plotted using matplotlib [3]. The mean wind speed for many weather summaries is displayed on the bar graph. "Clear", "Foggy", and "Overcast" are the three classifications shown under the "Summary" x-axis label. The scale runs from 0 to 12, with the caption "Mean Wind Speed" on the y-axis. The data are not displayed. For every category in the weather report, bars are shown on the graph. This graph indicates that wind speed and cloud cover are inversely correlated. The days with the highest mean wind speed are clear ones with little cloud cover. On the other hand, the mean wind speed is lowest on cloudy days with the highest amount of cloud cover. The typical wind speed is moderate during foggy days.

Model training and testing:

The main aim is to build a model which predicts the daily summary with the help of factors like temperature, humidity, wind speed, wind bearing, visibility, pressure. Before building the model, I plotted a correlation Heatmap to check the correlation of the attributes.

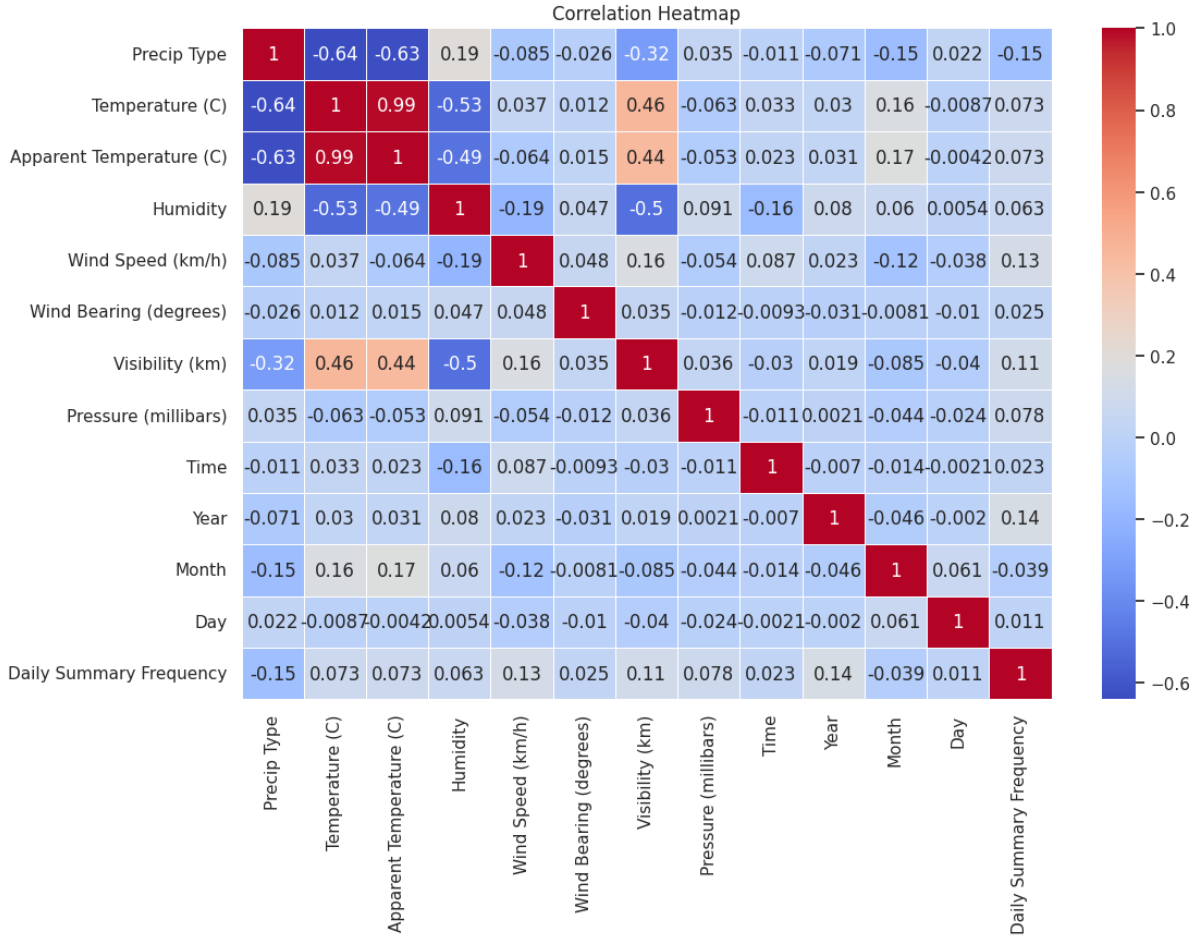

Figure 7: Correlation Heatmap (Jupyter Notebook)

The above graph figure 7 is plotted using matplotlib. The graph, which displays the correlation between several meteorological variables, is a correlation heatmap. Positive correlations are displayed in red on the heatmap, while negative correlations are displayed in blue. The color intensity in the heatmap indicates the strength of the link. Perfect positive correlation is represented by a value of 1, while perfect negative correlation is represented by a value of -1. A correlation of 0 means there is none.

I chose 4 models in-order to get the best results. The 4 models are

- Logistic Regression [13]: In order to fit a linear equation to the data, logistic regression first uses a sigmoid function to modify the result. The linear output between 0 and 1, which represents the likelihood that an observation belongs to the positive class, is squashed by this function. Simple to understand, performs well with linear data, and needs comparatively less training data than some other models. utilized in categorization issues when the answer is either true or false (yes/no). It forecasts the likelihood that an event will fall into a particular class.

```
Best Hyperparameters: {'C': 10.0, 'multi_class': 'ovr', 'penalty': 'l1', 'solver': 'liblinear'}

Testing Performance
Algorithm:  Logistic Regression

Accuracy: 0.795
Precision: 0.793
Recall: 0.795
F1 Score: 0.794
```

Figure 8: Accuracy of logistic Regression (Jupyter Notebook)

From the Logistic Regression model, the accuracy is got is 79.5%, the precision is 79.3% which is good but not the best among all the models.

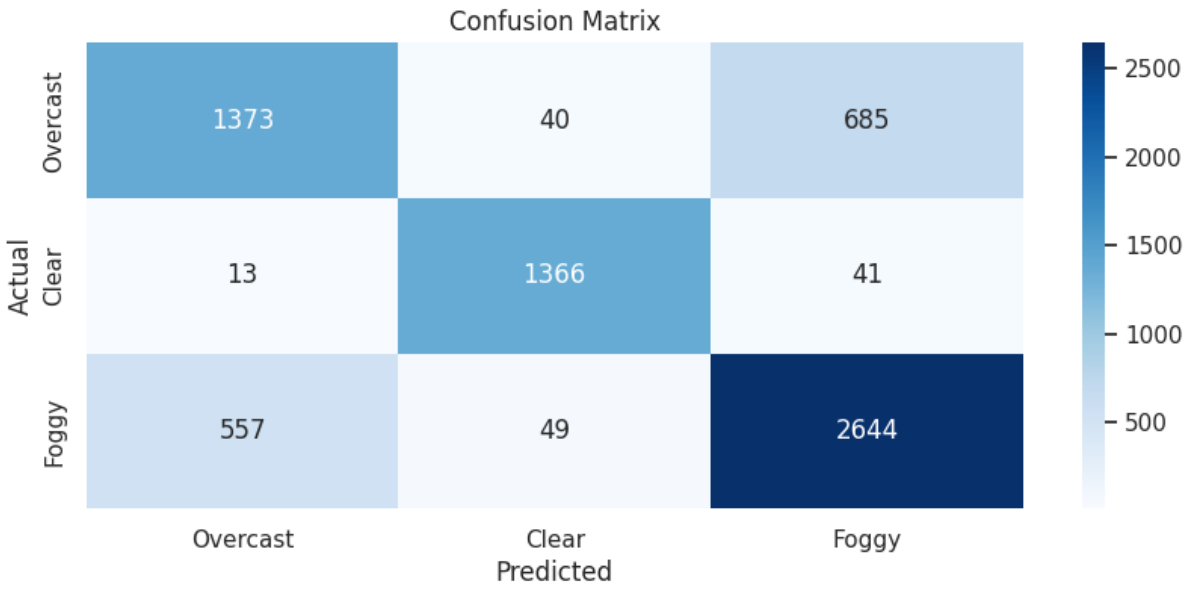

Figure 9: confusion matrix of actual and predicted values of logistic regression model(Jupyter Notebook)

The figure 9 is plotted seaborn library [6]. The table that represents the graph is called a confusion matrix, and it is used to show how well a classification model is performing. Here, the model is attempting to forecast the kind of weather—cloudy, clear, or foggy.

- KNN: KNN stores the training data in memory rather than creating a model manually. In order to forecast a new data point, it first locates the k closest neighbors in the training set, then uses the average value (regression) or most frequent class (classification) among those neighbors to assign the new data point. Easy to comprehend and apply, effectively handles high-dimensional data. activities including regression and

classification. KNN is a non-parametric technique that uses the similarity between each data point and its k nearest neighbors in the training set to classify the data points.

```
Best Hyperparameters: {'algorithm': 'auto', 'p': 1, 'weights': 'distance'}

Testing Performance
Algorithm:  KNN

Accuracy: 0.837
Precision: 0.837
Recall: 0.837
F1 Score: 0.837
```

Figure 10: Accuracy of KNN (Jupyter Notebook)

From the KNN model, the accuracy is 83.7% and precision is 83.7%. KNN model got high accuracy compared to Logistic Regression.

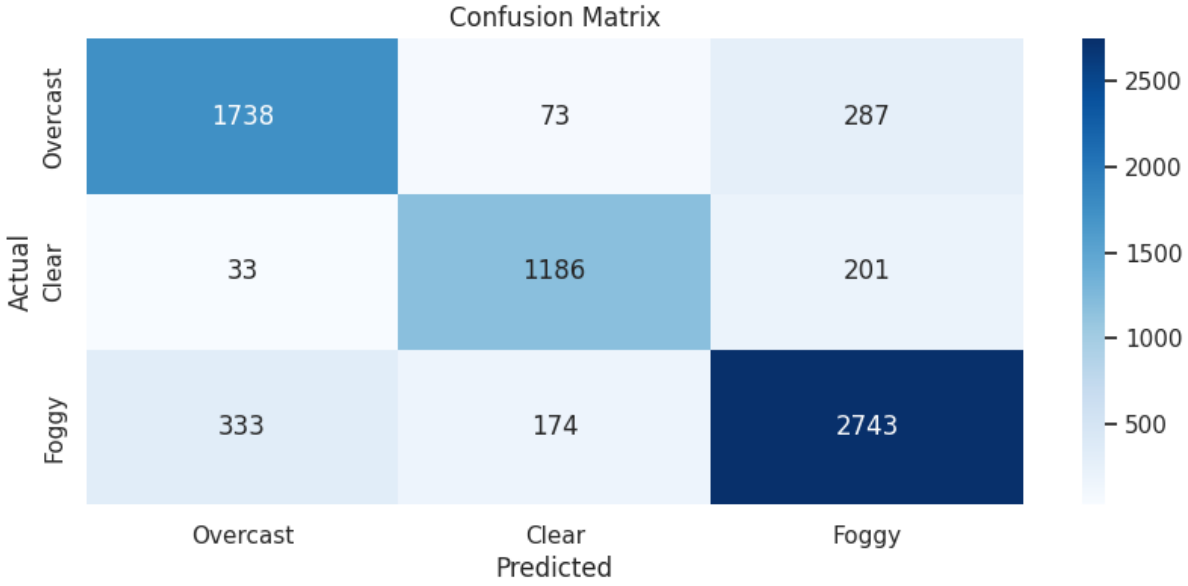

Figure 11: confusion matrix of actual and predicted values of KNN model (Jupyter Notebook)

The figure 11 is plotted seaborn library [6]. The table that represents the graph is called a confusion matrix, and it is used to show how well a classification model is performing. Here, the model is attempting to forecast the kind of weather—cloudy, clear, or foggy.

- Decision tree [14]: To arrive at a final forecast, the decision tree applies a sequence of if-then-else rules depending on the characteristics. The forecasts' reasoning is simple to understand and can handle both numerical and category variables. activities including regression and classification. By dividing the data into subgroups according to certain feature values, decision trees are trained. As a result, a structure resembling a tree is created, with each node standing for a decision point and the branches for many possible outcomes.

```
Best Hyperparameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'splitter': 'best'}

Testing Performance
Algorithm:  Decision Tree

Accuracy: 0.897
Precision: 0.896
Recall: 0.897
F1 Score: 0.896
```

Figure 12: Accuracy of Decision tree (Jupyter Notebook)

From Decision tree, the accuracy is 89.7% and precision is 89.6%. Decision tree got even more accuracy than KNN model.

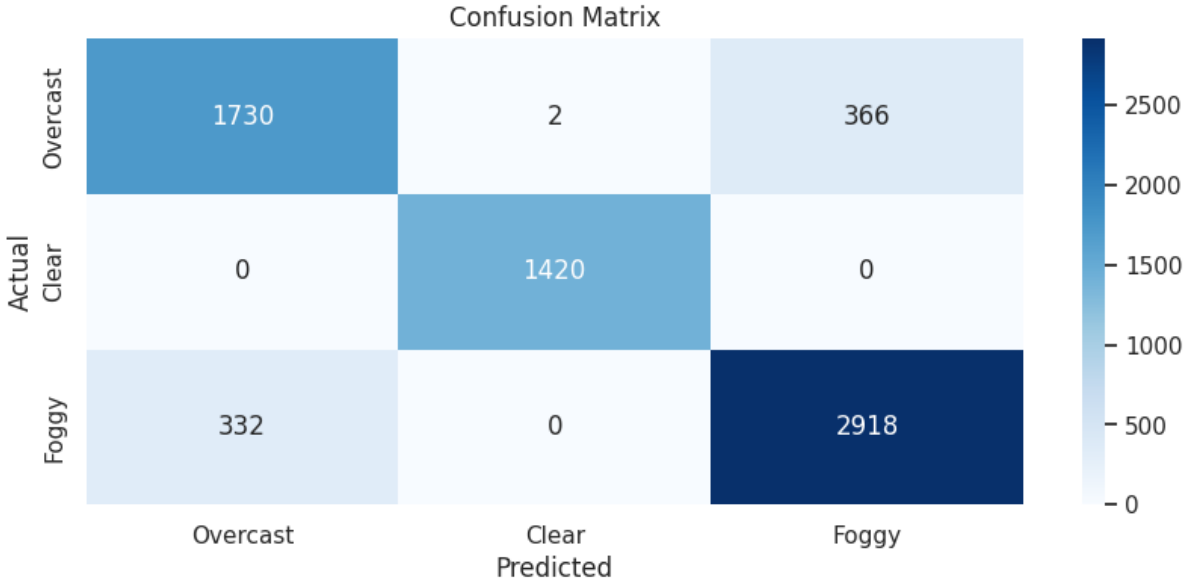

Figure 13: confusion matrix of actual and predicted values of decision tree (Jupyter Notebook)

The figure 13 is plotted seaborn library [6]. The table that represents the graph is called a confusion matrix, and it is used to show how well a classification model is performing. Here, the model is attempting to forecast the kind of weather—cloudy, clear, or foggy.

- RandomForest [15]: Compared to a single decision tree, random forests improve generalization and decrease overfitting by utilizing the combined strength of numerous decision trees. It handles both numerical and category variables and is typically more reliable and precise than individual decision trees. activities including regression and classification. Decision tree ensembles are called random forests. They function by using randomized selections of characteristics and data points to train multiple decision trees. The different trees' predictions are then combined to create a forecast (e.g., average for regression, majority vote for classification).

```
Best Hyperparameters: {'class_weight': 'balanced', 'max_depth': None, 'min_samples_split': 2}

Testing Performance
Algorithm:  Random Forest

Accuracy: 0.941
Precision: 0.941
Recall: 0.941
F1 Score: 0.94
```

Figure 14: Accuracy of RandomForest (Jupyter Notebook)

From RandomForest, the accuracy is 94.1% and precision is 94.1%. This model got the highest accuracy compared to the other three. RandomForest is the best model among the four.

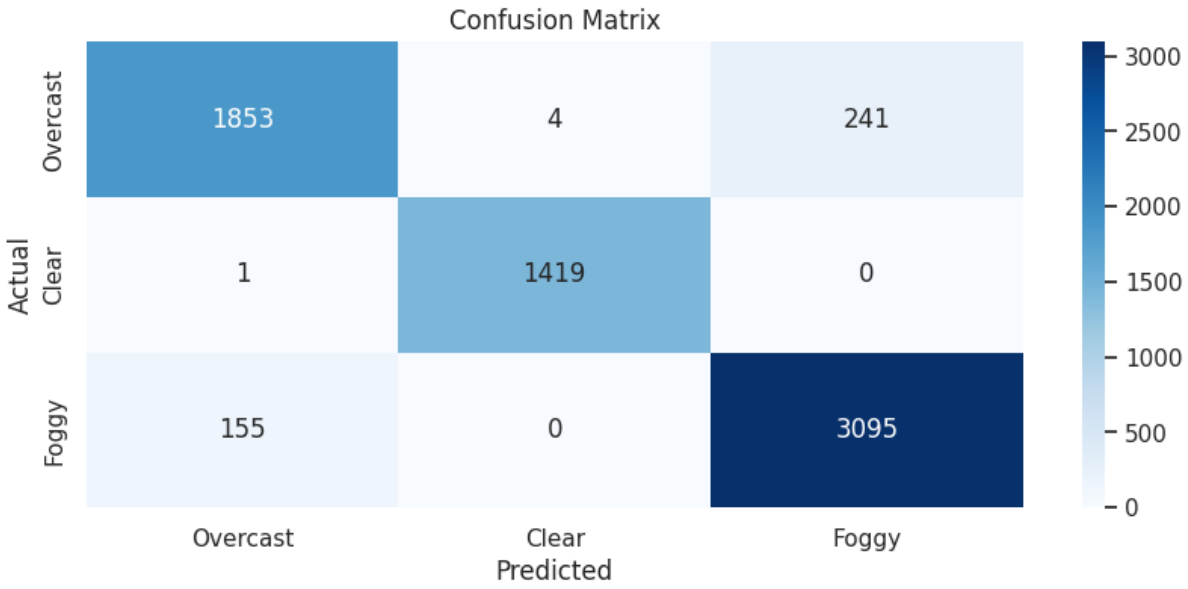

Figure 15: Confusion matrix of actual and predicted values of RandomForest (Jupyter Notebook)

The figure 15 is plotted seaborn library [6]. The table that represents the graph is called a confusion matrix, and it is used to show how well a classification model is performing. Here, the model is attempting to forecast the kind of weather—cloudy, clear, or foggy.

The 4 models are successfully designed and compared to know the best model that can predict the daily summary.

	Model	Accuracy	Precision	Recall	F1
0	Logistic Regression	0.795	0.793	0.795	0.794
1	KNN	0.837	0.837	0.837	0.837
2	Decision Tree	0.897	0.896	0.897	0.896
3	Random Forest	0.941	0.941	0.941	0.940

Table 3: Comparison of accuracies of 4 models (Jupyter Notebook)

The above table is plotted using pandas library [2]. From the above table, it is confirmed that RandomForest is best model.

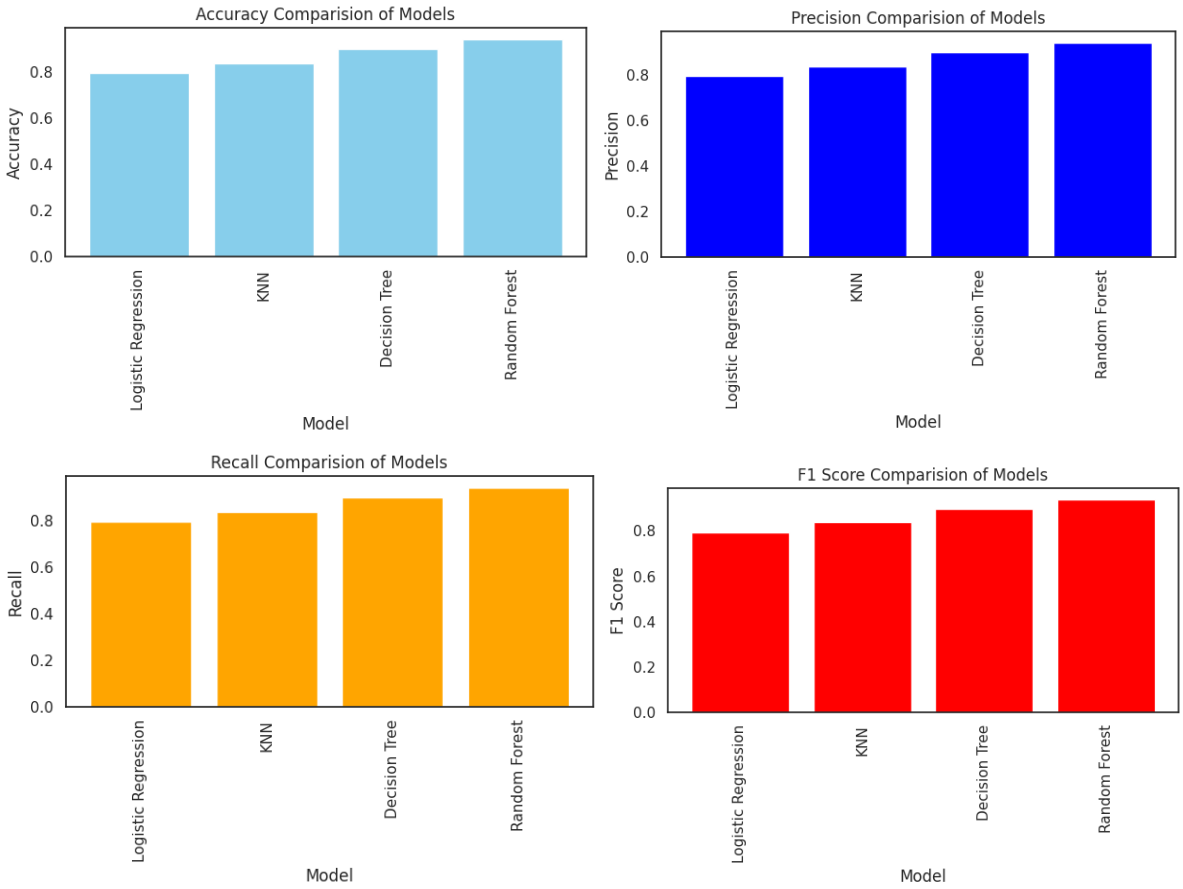

Figure 16: Bar graph comparison of accuracy, precision, recall, F1 score of 4 models (Jupyter Notebook)

The 4 graphs are plotted using matpotlib library [3]. The graphs show the comparison of 4 different attributes such as accuracy, precision, recall, F1 score of the 4 models. From the 4 models RandomForest got highest accuracy.

Effectiveness of the analysis:

From the model RandomForest I build, the accuracy is 94.1which is a good accuracy to predict daily summary with required information. From the exploratory analysis, I got the trend of pressure over time. The change of pressure is analysed with the graph. The other exploratory graphs helped in building the ML model.

My hypothesis description:

I hypothesized that there may exists a correlation between temperature and humidity. Yes, from my analysis I found it true. From figure 7 correlation Heatmap I found that Temperature has a strong positive correlation with Humidity which means the temperature increases, there is a tendency for the humidity to also increase, and vice versa. My aim is to build a model which predicts daily summary with good accuracy, the RandomForest gave me a good accuracy with good precision.

Lessons learned:

From the weather prediction analysis, I came to know the effectiveness of correlation matrix.

From the correlation matrix I plotted, I completed my hypothesis. I learned to build multiple models. I came to know the effectiveness of the RandomForest model.

Conclusion:

In the end, I build a model which effectively analyse the given data and give a daily summary report. The model got an accuracy of 94.1% which helps in getting the true results with very little number of errors. The analysis I made helped me in building the model with good accuracy. The modelling techniques, analysis methods made my project finish quickly and effectively.

Reference:

[1] Muthukumar (Ed), (2017). Whether Dataset, Kaggle. https://www.kaggle.com/datasets/muthuj7/weather-dataset/code

[2] Installation — pandas 2.2.2 documentation. (n.d.). Retrieved April 28, 2024, from https://pandas.pydata.org/docs/getting_started/install.html

[3] Installation — Matplotlib 3.8.4 documentation. (n.d.). Retrieved April 28, 2024, from https://matplotlib.org/stable/users/installing/index.html

[4] Project Jupyter. (n.d.-b). Installing Jupyter. https://jupyter.org/install

[5] Anaconda. (2024c, April 23). *Unleash AI Innovation and Value | Anaconda V2.4.2*. https://www.anaconda.com/

[6] Installing and getting started — seaborn 0.13.2 documentation. (n.d.). Retrieved April 28, 2024, from https://seaborn.pydata.org/installing.html

[7] Installing NumPy — NumPy v1.26 Manual. (n.d.). Retrieved April 28, 2024, from https://numpy.org/install/

[8] datetime — Basic date and time types. (n.d.). Python Documentation. Retrieved April 28, 2024, from https://docs.python.org/3/library/datetime.html

[9] Installing joblib — joblib 1.4.0 documentation. (n.d.). Retrieved April 28, 2024, from https://joblib.readthedocs.io/en/stable/installing.html

[10] Installing scikit-learn. (n.d.). Scikit-learn. Retrieved April 28, 2024, from https://scikit-learn.org/stable/install.html

[11] Install TensorFlow 2. (n.d.). TensorFlow. Retrieved April 28, 2024, from https://www.tensorflow.org/install

[12] wordcloud. (2023, December 9). PyPI. Retrieved April 28, 2024, from https://pypi.org/project/wordcloud/

[13] Cox, D. R. (1958). The regression analysis of binary sequences. Journal of the Royal Statistical Society: Series B (Methodological), 20(2), 215–232.

[14] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., … others. (2008). Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1), 1–37.

[15] Ho, T. K. (1995). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278–282).